IN THE SPECIFICATION:

Please amend page 13, lines 16-18 as follows:

S = C[0]*C[1]*...*C[511] mod N

= (y^(x[0]*4^511))*(y^(x[0 1]*4^510))*...*(y^x[0511]) mod N

= y^(x[0]*4^511 + x[1]*4^510 +... + x[511]) mod N

= y^x mod N

Please amend page 19, lines 6-10 as follows:

Herein, 160-bit scalar multiplication computation is performed. It is known that a key length of 160 bits in the elliptic curve encryption corresponds to a key length of 1024 bits in the RSA encryption. Hereinafter, binary representation of k is written as (k[0]k[1] ... k[79]), where k[j] ~~x[j]~~ is a 2-bit block that is equal to one of 00, 01, 10, and 11.

Please amend page 21, last paragraph (continuing to page 22) as follows:

FIG. 13 shows a processing for creating a table required for computation. First, S is initialized to y (1001), and a counter j is set to 512 (1002). Next, an end condition is judged in a conditional branch processing 1003. Next, S is stored in RAM (1004). (Hereinafter, when this value is written as B[j] for the counter j, B[j]s are placed such that they do not overlap with each other on the memory, as in the processing of FIGS. 5 to 7.) Next, the embodiment calculates S to the fourth power modulo N in 1005, the counter j is decremented in 1006, and the control returns to the conditional branch processing 1003 again. After this operation is repeated 512 times, the control transfers to processing 1007 of FIG. 14. In 1007, the condition for the counter j is judged again. This is an end judgment. If j is not 512, the processing proceeds to 1008. In 1008, j-th value V(j) is read from an EEPROM area PERM_tmp for random permutation as shown in FIG. 16. As shown in FIG. 16, V(j)s are stored beforehand with a random rearrangement of 0, 1, 2,..., 511. Next, in 1009, V(j) is rewritten to V(j)=((17*(V(j)+1)) mod 513)-1. (Although replacement is made for each j in this embodiment, some EEPROMs permit only replacement in pages, in which case V(j) of each page size should have been stored in RAM to make a collective replacement.) Operation of this portion will be described from a more common standpoint.

Please amend page 23, lines 9-18 as follows:

S = C[V(0)]*C[V(1)]*...*C[V(511)] mod N

= (y^(x[V(0)]*4^(511 - V(0)))*(y^(x[V(1)]*4^(512 - V(1)))* ...*
(y^(x[V(0511)]*4^(511 - V(511))) mod N
= y^(x[V(0)]*4^(511 -V(0)) + x[V(1)]*4^(512 - V(1)) + ...
+ x[V(0511)]*4^(511 - V(511)) mod N
is satisfied.

Because of the nature of mapping V, since V(0), V(1), ..., V(511) is an rearrangement of 0, 1, ..., 511, the above described exponent part x[V(0)]*4^(511 - V(0)) + x[V(1)]*4^(512 - V(1)) + ... + x[V(0511)]*4^(511 - V(511)) is equal to x[0]*4^511 + x[1]*4^510 + ... + x[511]. Accordingly, S is equal to y^x mod N.

Please amend page 24, last paragraph (continuing to page 25) as follows:

FIG. 17 shows a processing for creating a table necessary for computation. S is initialized to a base point P (1101), and a counter J is set to 80 (1102). Next, an end condition is judged in a conditional branch processing 1103. Next, S is stored in RAM (1104). (Hereinafter, when this value is written as B[j] for the counter j, and B[j]s are placed such that they do not overlap with each other in the memory, as in the processing of FIGS. 5 to 7.) Next, S is multiplied by four on an elliptic curve E in 1105, the counter j is decremented in 1106, and the control returns to the conditional branch processing 1103 again. After this operation is repeated 80 times, the control transfers to a processing 1107 of FIG. 18. In 1107, the condition for the counter j is judged again. This is an end judgment. If j is not 80, the processing proceeds to 1108. In 1108, j-th value V(j) is read from an EEPROM area PERM_tmp for random permutation as shown in FIG. 20. As shown in FIG. 20, V(j)s are stored beforehand with a random rearrangement of 0, 1, 2,..., 79. Next, in 1109, V(j) is rewritten to V(j)=((7*(V(j)+1)) mod 81) - 1. (Although the replacement is made for each j in this embodiment, some EEPROMs permit only replacement in pages, in which case V(j) of each page size should have been stored in RAM to make a collective replacement.) This operation rearranges {0, 1, 2, 3, ..., 79}, as shown previously in the embodiment for the RSA encryption processing. Here, the number of 7 is only an example, and any number prime to 81 may define a permutation of {0, 1, 2, ..., 81}. Although other methods for generating a permutation are available, their description is omitted to avoid departing from the spirit of the present invention.

Please amend page 26, lines 6-12 as follows:

C[V(j)] = ~~B[V(j)]^x[V(j)]~~ k[V(j)]*B[V(j)] mod N(j = 0, 1, 2, ..., 511),

upon termination of all processings, the following expression

S = C[V(0)] + C[V(1)] +...+ C[V(511)]

= (k[V(0)]*4^(79-V(0))P + (k[V(1)]*4^(~~512~~79-V(1))P + ... + (k[V(~~0~~79)]*4^(~~511~~79 - V(~~511~~79))P

= (k[V(0)]*4^(79-V(0)) + k[V(1)]*4^(79-V(1)) + ... + k[V(~~0~~79)]*4^(79 -V(79)))P

is satisfied.

Because of the nature of mapping V, since V(0), V(1), ..., V(79) is an rearrangement of 0, 1, ..., 79, the above described scalar part k[V(0)]*4^(79 - V(0)) + k[V(1)]*4^(79 - V(1)) + ... + k[V(~~0~~79)]*4^(79 - V(79)) is equal to k[0]*4^79 + k[1]*4^78 + ... + k[79]. Accordingly, S is equal to kP.

Please amend page 27, lines 3-15 as follows:

(2^m ± 1)P = (2^m)P + (-P).

For example, although 195 can be written as 11000011[[1]] in binary notation, it is split to 2-bit blocks so as to be written as (11,00,00,11). Since 11 is first encountered during examination from the lowest-order position, 1 is added to a higher-order position of 11 to make (11,00,01,11). 11 in the highest-order position is again added with 1 in higher-order positions thereof to make (01,11,00, 01,11). In such conversion, 11 is read as -1 for interpretation. That is, the number 195 is represented as follows:

Before conversion: 195 = 3*4^3 + 0*4^2 + 0*4 + 3

After conversion: 195 = 1*4^4 + (-4^3) + 0*4^2 + 1*4 + (-1).

This is equivalent to a rewrite to 3 = 4 ± 1. This is applied to scalar multiplication operation as follows:

195P = 1*((4^4)P) ± (4^3)P + 0*((4^2)P) + 1*4P - P.

Please amend page 28, last paragraph as follows:

The following processing is performed after the above preparations have been made. First, according to processing of FIG. 21, ~~P[j]~~ B[j]=(4^j)P(j = 0, 1, 2, ..., ~~80~~79) is computed, and a lookup table is created. S is initialized to a base point P (1201), and a counter ~~I~~ j is set to 80 (1202). Next, an end condition is judged in a conditional branch processing 1203. Next, S is stored in RAM (1204). (Hereinafter, this value is written as B[j] for the counter j, and B[j]s are placed such that they do not overlap with each other in the memory, as in the processing of FIGS. 5 to 7.) Next, S is multiplied by four on an elliptic curve E in 1205, the

counter j is decremented in 1206, and the control returns to the conditional branch processing 1203 again. After this operation is repeated 80 times, the control transfers to processing 1207 of FIG. 22. In 1207, the condition for the counter j is judged again. This is an end judgment. If j is not 80, the processing proceeds to 1208. In 1208, j-th value V(j) is read from an EEPROM area PERM_tmp for a random permutation as shown in FIG. 20. As shown in FIG. 20, V(j)s are stored beforehand with a random rearrangement of 0, 1, 2,..., 79. Next, in 1209, V(j) is rewritten to V(j)=((7*(V(j)+1)) mod 81) -1. This operation rearranges {0, 1, 2, 3, ..., 79}, as shown previously in the embodiment for the RSA encryption processing. Of course, the number of 7 is only an example, and any other number prime to 81 may define a permutation of {0, 1, 2, ..., 81}. Although other methods for generating a permutation are available, their description is omitted to avoid departing from the spirit of the present invention.